Mission Planning for a Robot Factory Fleet

Mission Planning for a Robot Factory Fleet

Matthew Crosby[1] and Ronald P. A. Petrick[1]

***Abstract*— Planning is becoming increasingly prevalent as a tool for high-level reasoning in real-world robotics systems. This paper discusses the implementation of a high-level 'mission planner' that utilises AI planning techniques to find initial plans for a fleet of robots acting in a manufacturing factory. The paper introduces the system architecture, and then focuses on the ROS-based mission planning component, which requires the translation of low-level robot 'skills' and a world model to a high-level planning domain. This paper also introduces a new algorithm for decomposition-based planning that can find 'balanced' plans in large multi-robot domains where current state-of-the-art techniques fail.**

I. INTRODUCTION

Robots have been used effectively in factories for many years, however, it is only recently that they have begun to take on roles requiring large amounts of autonomy and high-level reasoning capabilities. Autonomy is becoming increasingly necessary due to greater variability in factory-generated products (e.g., end-user customisation) leading to assembly lines that must involve more than just simple repetitions of the same task with the same components. For example, in car manufacturing—the application domain for this work—each car is built to the specification of the user, meaning that the parts required for assembly may be different for each car on the line. Not only does this require more sophisticated assembly line robotics, but robots are starting to be utilised in preparing the parts for delivery to the assembly line. The proposed solution in this paper is to use autonomous mobile robots with the ability to pick parts from around the factory and place them in a *kit* to be delivered to the assembly line at the appropriate time (see Fig. 1).

In our application domain, a centralised controller, called the *mission planner*, is tasked with creating and assigning initial high-level plans to the robots in the fleet. These initial plans are passed to the robots as input for the on-board robot task planners. Given that the robots can act in the environment concurrently, it is desirable that plans are evenly distributed amongst the robots so as to reduce total execution time. Furthermore, each robot in the domain is capable of filling multiple kits at a time and it is the mission planner's job to work out an efficient allocation of kits to robots, as well as to generate the high-level actions needed to fill these kits without obstructing the other robots.

The robot fleet utilises a skills framework which modularises robot capabilities into high-level, symbolic planning actions, meaning that the robots are pre-calibrated to use planning-like constructs. The generated plans are produced in a standard planning format that is used as input to the on-board robot task planners (responsible for reasoning about sensing actions and manage re-planning activities) so that execution can begin immediately. This setup allows the technology to be reused when robot capabilities and the environment change, as happens when the factory floor is rearranged, new parts become available, or new robots are added to the fleet.

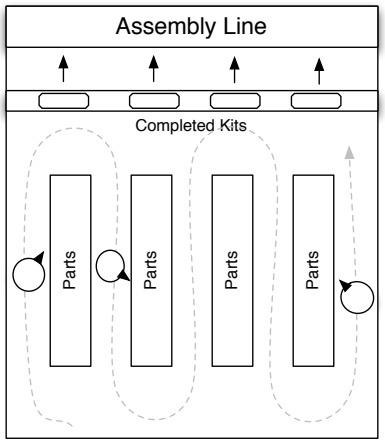

Fig. 1. The factory environment. Multiple autonomous mobile robots navigate a warehouse preparing kits to be brought to the assembly line.

For our application domain it was found that current planning techniques were either too slow (when dealing with multiple robots) or that the output plans did not have an acceptable makespan (i.e., were not distributed efficiently amongst the robots). The main contribution of this paper is a planning algorithm called ADBP that builds on previous work to allow for low makespan plans to be found in a time-frame sufficient for use in the factory.

II. BACKGROUND

Planning has a long association with robotics, stretching as far back as Shakey [1] and Handey [2]. However, the use of planning has often centred around robot-level task planning which, while abstracted from low-level robot control, still needs to deal with contingencies, sensing actions, and other robot-level considerations. This work considers a problem that is abstracted even further: while it avoids some of the complexities that arise in typical robotics planning problems, it also introduces new complications (e.g., multiple robots).

A key component of the framework for the application of this work is the concept of *robot skills* [3]. Robot skills

The research leading to these results has received funding from the European Union's Seventh Framework Programme under grant agreement no. 610917 (STAMINA), http://stamina-robot.eu/.

[1]The authors are at the School of Informatics, University of Edinburgh, Edinburgh, EH8 9AB, Scotland, United Kingdom. Email: m.crosby@ed.ac.uk, rpetrick@inf.ed.ac.uk

are designed to bridge the gap between robot-level control operations and planning-level actions, by providing a structured representation of both the requirements (preconditions) and expected outcomes (effects) of a robot-level action, encapsulated together with a set of methods for verifying these conditions in the real world through sensing. Thus, skills not only provide a model of preconditions and effects, but also the ability to evaluate and verify these conditions.

The particular skills framework we use provides a standard STRIPS-style PDDL encoding of the domain [4], [5], enabling us to consider utilising successful off-the-shelf planners such as LAMA [6] and FF [7] which support this representation. However, while these planners perform well on small domain instances with two or less robots, they do not scale well with the number of robots.

Another approach to this problem is to use a temporal planning encoding with the features introduced in PDDL 2.1 [8]. Temporal planning allows for easy use of numeric fluents to simplify the encoding, and can be used with planners such as POPF2 [9], a forward chaining partial-order planner that was the runner up in the temporal track of the 2011 International Planning Competition (IPC) [10]. POPF2 automatically attempts to minimise makespan, returning well-balanced plans. Unfortunately, temporal planning does not scale any better than standard single-agent approaches with respect to the number of robots.

Given the previous results, this work instead focuses on multiagent techniques that attempt to exploit the underlying structure inherent in multiagent domains [11]. Due to the centralised nature of the problem, this approach circumvents common multiagent planning challenges such as decentralised planning, strategic elements, concurrency constraints for joint actions, and privacy concerns.

During the course of this work, we tested multiple multiagent planners and found that none could solve problems of the size required for our application: MA-FD [12] could not scale past two robots on our full size problem; MAPR [13] could not scale past 4 agents when using *load-balance*, a goal assignment strategy which attempts to keep a good work-balance among the agents; and both ADP and [14] and MAPR could solve the problem, but only if allowed to return a single-robot solution (i.e., one robot was assigned to do all the tasks regardless of the number of robots in the domain). Overall, the current techniques we tested could only scale beyond a few robots if they ignored the multiagent nature of the problem. This paper therefore focuses on showing it is possible to find truly multiagent plans in this domain that consider all the robots.

III. MISSION PLANNER AND SYSTEM ARCHITECTURE

Fig. 2 gives a simplified overview of the system architecture we use for this work. The architecture contains multiple robots, each with an on-board *task planner* for real-time planning and replanning, and a *skill manager* that handles the translation of low-level robot control nodes (*skills*) to the high-level parts of the system. The *logistics planner* acts as a system manager that controls the information and world model for the whole system and supplies the input connections for the goals and data provided at the factory level. The mission planner takes its inputs about goals and skills from the logistics planner and returns multi-robot plans which are, in turn, passed to the appropriate robots.

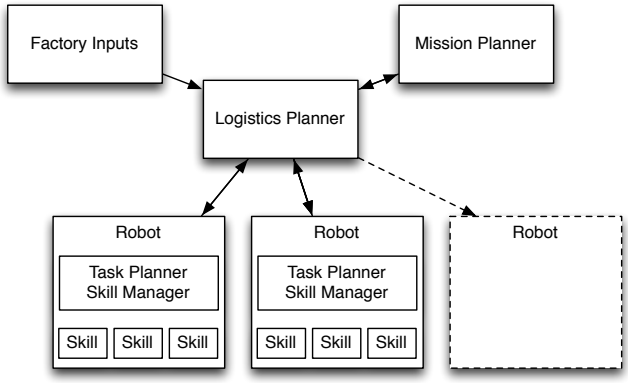

Fig. 2. System architecture diagram.

```
001: NAVIGATE(robot1, startloc, kit1)
002: GETKITBOX(robot1, kit1)
003: NAVIGATE(robot1, kit1, kit2)
004: GETKITBOX(robot1, kit2)
005: NAVIGATE(robot1, kit2, loc4)
006: PICK(robot1, part1, box4)
007: PLACE(robot1, part1, kit1)
008: NAVIGATE(robot1, loc4, loc12)
009: PICK(robot1, part5, loc12)
010: PLACE(robot1, part5, kit2)
...
035: NAVIGATGE(robot1, loc45, locoutput)
036: PLACEKIT(robot1, kit1, outputconveyor)
037: PLACEKIT(robot1, kit2, outputconveyor)
```

Fig. 3. An example plan in which `robot1` completes two kits and places them on the output conveyor.

The mission planner's inputs are a copy of the current world model, the skills available to each robot, and the current goal set. The goal set is comprised of an ordered list of kits that must be assembled, where each kit contains a list of parts that need to be picked. The mission planner outputs a plan for each robot consisting of an ordered list of instantiated skills. An example single-robot plan is shown in Fig. 3. In this plan, `robot1` fills two kits and places them on the output conveyor. In the remainder of the paper we focus solely on the mission planner component.

IV. PLANNING DOMAIN AND ENCODING

To test the high-level planning component of the system, a complete version of the warehouse domain was implemented and tested offline with multiple robots and hundreds of parts.[1] (See Fig. 4 for a pictorial representation of the domain

[1]The current system has also been tested online in a small domain, however, due to the limitations of real-world testing, this domain was restricted to a single robot with only a few parts available. The work in this paper instead focuses on potential future applications of this domain with multiple robots which need to handle all the parts in the factory.

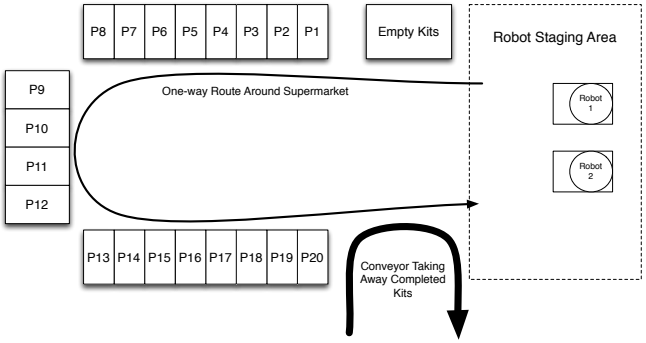

Fig. 4. The warehouse testing domain. Robots must follow the track around the warehouse and cannot overtake other robots. Parts are stored in labelled boxes, and must be appropriately selected to complete a kit. Completed kits are placed on the conveyor. Robots may carry up to two kits at a time and may only pass one another in the staging area.

setting.) In this domain, the robots cannot overtake each other except in the staging area (in which case they can return to their base and then leave before another robot). Each robot has space to carry up to two kits at a time, where a kit is a box with a number of compartments, each designed to fit a specific part. The aim of the mission planner is to provide initial plans for each of the robots in the fleet so that they can complete their task of assembling kits within a specific time frame and without conflicting with other robots.

For the encoding, we model the robots, the position of the kits on the robots, the kits themselves, the available parts, and the locations in the factory. From the skills framework, we are given that robots can perform the actions `NAVIGATE`, `PICK`, `PLACE`, `GETKIT`, and `PLACEKIT`.

One interesting aspect of the domain encoding concerns how to determine when a kit is full and ready to be placed on the conveyor. Multiple approaches can be considered, such as: 1) numeric fluents that count the number of parts in a kit and actions with conditional effects that consider whether a kit was full, 2) a 'close-kit' action that can only be performed when the numeric fluents count the correct number of parts in a kit, 3) both of the above approaches, except with the numeric fluents encoded in number objects with a successor relation, and 4) high-arity predicates or actions with a large number of parameters, coupled with either equality constraints (one for each possible pair of parts), or a careful encoding to ensure parts are guaranteed to be different. For example, a high-arity 'deliver-kit' action requiring 6 parts might include the parameters:

```
(?r - robot ?l - location ?k - kit
    ?p1 - part ?p2 - part ?p3 - part
        ?p4 - part ?p5 - part ?p6 - part)
```

Testing found that none of the above encodings allow for solutions to larger problem instances with any of the planners. One solution that works is to encode kit information as part of the goal for each problem, with separate propositions denoting 'delivered(kit)' and 'in-kit(part)'. While, in general, large goal sets are a problem in planning (e.g., the visit-all domain [15]), they avoid the need for high-arity predicates or parameters, numeric fluents, and inequalities, and are the only method that allowed the temporal and single-agent planners to solve even small domain instances. goal distribution during search performed by ADP.

Fig. 5 shows an overview of the mission planner's operation from input to output. A world model, set of skills, and current goals as used as input. The planning domain is created based on a skills list using a library of precomputed skill-to-PDDL conversions. This list may change depending on the active robots in the domain and the skills they possess.

When creating the problem file for a particular planning instance, the planner does not code any unnecessary information. For example, if a part does not appear in any of the goals then it does not need to be modelled by the planner. Therefore, when creating the problem file, the mission planner iterates over the parts that occur in the goals, looks up their locations and properties in the world model and adds them to the problem instance. Similarly, locations that only contain parts that do not appear in the goals are not included in the domain.

The planning domain and problem are used as input to a planner that returns a single multi-robot plan. The multi-robot plan is then broken down into a plan for each robot by separating out each agent's actions while maintaining the original plan ordering.

To create more efficient planning domains, the planning problems are modified such that the plans need to be pre-processed before being returned as actionable plans to the logistics planner. For example, the world model contains a total strict ordering of all the locations in the domain that respects the route the robots must take around the warehouse. This information is encoded using a `BEFORE` predicate, such as `BEFORE(loc1,loc2)`, and the `NAVIGATE` action can only be used to navigate from `locx` to `locy` if such a predicate exists for each point. In practice, we found that planners perform better if we encode this relationship only for adjacent locations so that, for example, `BEFORE(loc1,loc4)` does not exist. Instead, the robot will have to `NAVIGATE` from `loc1` to `loc2`, then `loc2` to `loc3`, and so on, until it reaches `loc4`. Any time two (or more) `NAVIGATE` actions appear in a plan, they are concatenated to form a single `NAVIGATE` action. From the previous example, the three `NAVIGATE` actions would be concatenated to form `NAVIGATE(loc1,loc4)`, before being sent to the logistics planner.

V. The ADBP Algorithm

Initial tests (see Table I) showed that ADP and MAPR were the best performing planners as the number of robots in the domain scaled, however, they could only return single-robot plans which are not helpful for practical purposes. No planners that could return low makespan plans could scale beyond problem 4 (this also includes MA-FD which could solve 1 and MAPR-lb which could solve 4). We now describe ADBP (Agent Decomposition Balanced Planner) which can solve up to problem 10 in under 10 seconds, and returns multi-robot solutions.

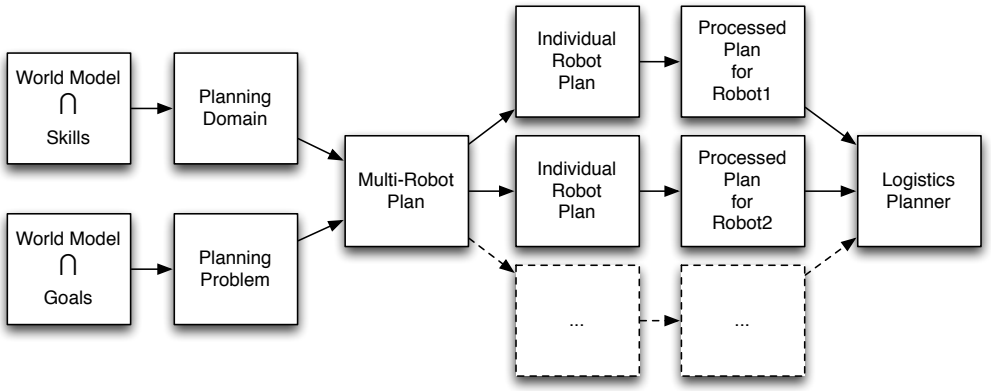

Fig. 5. An overview of the mission planning process from input to output and back to the logistics planner.

TABLE I

TABLE SHOWING THE PERFORMANCE OF PLANNERS ON TESTING DOMAINS AS THE SIZE OF THE PROBLEM INCREASES TO THE SIZE REQUIRED FOR APPLICATION IN THE REAL-WORLD DOMAIN. RESULTS SHOW THE TIME IN SECONDS, THE TOTAL COST, AND THE MAKESPAN WHEN THE OUTPUT IS CONTRACTED TO A MULTIAGENT PLAN WITH JOINT ACTIONS. A '–' MEANS THAT THE PLANNER DID NOT RETURN A PLAN WITHIN 300 SECONDS.

Problem	No. of	No. of	POPF2			FF			LAMA			ADP		
Number	Robots	Goals	time(s)	cost	ms	time(s)	cost	ms	time(s)	cost	ms	time(s)	cost	ms
1	2	4	0.52	29	13	0.01	30	18	0	24	12	0.01	24	24
2	2	4	–	–	–	0.34	54	27	0.01	54	27	0	54	54
3	2	4	–	–	–	0.01	74	37	0.01	74	37	0.01	74	74
4	2	6	–	–	–	7.11	132	74	0.04	111	74	0.02	132	132
5	4	6	–	–	–	–	–	–	–	–	–	0.02	111	111
6	4	8	–	–	–	–	–	–	–	–	–	0.03	148	148
7	6	8	–	–	–	–	–	–	–	–	–	0.05	148	148
8	6	10	–	–	–	–	–	–	–	–	–	0.06	185	185
9	8	10	–	–	–	–	–	–	–	–	–	0.08	185	185
10	10	10	–	–	–	–	–	–	–	–	–	0.11	185	185

Algorithm 1: Heuristic Value Calculation of ADBP

Input : State S, Goals G
Output: h_max, h_square, or h_total

```
Relaxed Planning Graph Generation (full)
if Max layer > 0 then
    Calculate Subgoals
    G ← G ∪ subgoals
foreach agent i do
    foreach Goal g ∈ G do
        if h_add(g, i) > 0 then
            ExtractRelaxedPlan(g, i)
    hff(i) ← RelaxedPlan(i).cost
h_max = max_i hff(i)
h_square = Σ_i hff(i)^2
h_total = Σ_i hff(i)
```

ADBP provides a heuristic value for a given state and is implemented as a derived class of the Heuristic class in Fast Downward [16]. This means that ADBP can be used as a heuristic for different planning search methods, although we found that it was best suited to greedy best-first search.

ADBP uses the same agent decomposition as ADP which partitions the variables of the MPT representation of the problem created by Fast Downward such that each variable either belongs to an agent or to the environment [17]. We found that in all configurations and encodings of the domain tested this decomposition returned the expected partitioning of variables that picked out those that correspond to the robots in the domain.

ADBP's heuristic value calculation is shown in Algorithm 1. Relaxed Planning Graph (RPG) Generation is based on that introduced by FF [7], but modified for the multiagent case. Each agent generates its own RPG based on the problem reduced to only their variables and environment variables. After this, the union of the final state of each agent is then used to create another layer of RPGs. This process repeats until no further propositions can be added by any agent and guarantees that the full relaxed plan space is explored (see [17] for details).

If more than one layer of RPGs have been generated, then subgoals are calculated. Any time a goal proposition appears for the first time in a layer above the first, this means that it cannot be reached by an agent on its own. Therefore, plan extraction [7] is used to find out which proposition is utilised from the previous layer (see [17] for details). All necessary propositions from the first layer are added as subgoals.

In the final step of the heuristic calculation, each agent creates a relaxed plan to every goal in the goal set (including subgoals) that it can achieve. The value $hff(i)$ is the cost of agent i's relaxed plan. This only counts each action once

(even if it is used to reach multiple goal propositions).[2]

A. Heuristic Values

There are three different versions of ADBP, based on the way the individual heuristic values calculated by each agent at each state are combined:

ADBP-max uses the value h_max which is the maximum hff value of all the agents. This discards a lot of the information that has been calculated, but is as close as possible to the estimated makespan of the plan from the current state. As ADBP is primarily concerned with minimising makespan, this is a natural heuristic to investigate. The downsides are that it discards a lot of potentially useful information and has little concern for how close the goal state actually is. For example, 5 agents with $hff = 9$ return a worse heuristic value than if four have $hff = 0$ and one has $hff = 10$.

ADBP-total takes the other extreme and uses the sum of each agent's hff values. This is much further from the makespan heuristic estimate but encodes more information about the distance from the state to the goal. It should be noted that this is different from the single-agent FF value for the state because goals are repeated by all agents that can achieve them and subgoals are included in the calculation. The downside of this calculation is that it does not take the makespan into account at all (beyond the fact the value is calculated over multiple agents).

ADBP-square attempts to sit in the space between the two extremes and uses the sum of squares of the hff values of the agents. The idea is to use all the information available whilst taking the variance between the agent values into account.

VI. Evaluation

The evaluation is split into three parts. The first part discusses the performance of existing state-of-the-art planners on the factory domain, the second part explores ADBP's performance on the factory domain, and the final part briefly shows ADBP's performance in other multiagent domains.

Initial Results: The complete version of the domain was modelled with 10 robots, 10 goals, and a kit size of 6, however, this problem was not solvable by most of the planners in our initial tests. We therefore created smaller problem instances for testing purposes by varying the number of agents and goals. Each problem contains the full number of locations and parts, and parts-per-kit, except for two problems (0 and 1) which were smaller. It can be seen that the limiting factor is related to the number of agents and, once they are increased above 4, then only single-agent solutions can be found.

Along with time and cost, the results also show the makespan of the returned plans. For POPF2, this is simply the makespan as the temporal domain was designed with a direct mapping from actions at a time step in the temporal plan to joint actions. For the other approaches, the makespan can be calculated manually by post-processing plans using the algorithm presented in [18]. For our particular domain, the highest number of actions assigned to a single agent is a good enough approximation of the final makespan, and there was no need to calculate the exact joint plan at this stage. We use this approximation in the results reported in this paper.

ADBP Results: Table II shows the results for the different versions of ADBP on the factory domain (for the same problems as are shown in I). The first thing to note is that the makespans of all the ADBP algorithms are significantly lower than those for the original ADP. All the versions achieve (to some extent) the goal of distributing the plans amongst the agents. The table also shows that ADBP-max is not able to solve any problem beyond number 3. However, this is a problem with four robots, so there is still an improvement over the single-agent approach. This is perhaps somewhat surprising given the apparent lack of information contained in the heuristic value.

The most interesting feature about the results for ADBP-max is not shown in the table, but in the plans it outputs. Both ADBP-total and ADBP-square output plans in which the robots perform many actions in a row, normally completing a goal before another agent moves. The plans returned by ADBP-max contain actions that are interleaved between the different robots: the second agent follows directly behind the first as they navigate the factory. Given that we can compute a reduced makespan plan with post processing, this may not seem important, but the result shows an area for future work where interleaved actions may be more significant.

Multiagent IPC Domains: ADBP was also tested on a set of multiagent IPC domains [15], the results of which are shown in Table III. The results show that ADBP is not competitive in terms of planning speed: the results are given for problem 10 of each set, with ADBP not scaling well to the larger problems. However, it is interesting that it can return plans at all for these domains when considering every agent in each heuristic calculation, and it does achieve reasonable makespans. It should be noted that the behaviour of ADP-max was repeated an it returned interleaved plans on the smaller domains that it could solve.

VII. Conclusion and Future Work

This paper presented a mission planner for autonomous mobile robots operating in a factory environment, using a new decomposition-based planning algorithm that returns balanced multiagent plans. The proposed algorithm improved the quality of the returned plans, whilst still finding plans within the allotted time constraints for the application domain. Variations of the algorithm also exhibit interesting behaviour such as the interleaving of agent's actions, suggesting that this type of approach is promising for future work in multiagent environments.

We also tested our algorithm on multiagent IPC domains, but found that it is not competitive in its current instantiation. However, it is interesting to observe that ADBP works at all for domains with different properties to the warehouse domain. This indicates that the heuristic function is returning

[2]Note that ADBP supports preferred operators. Any action that appears in a relaxed plan (of any agent) that is also applicable in the current state is set as a preferred operator.

TABLE II

TABLE SHOWING THE PERFORMANCE OF THE DIFFERENT VERSIONS OF ADP ON THE TESTING DOMAINS. KEY: ORIG IS THE ORIGINAL ADP ALGORITHM. MAX IS ADBP-MAX, SQU IS ADBP-SQUARE, AND TOTAL IS ADBP-TOTAL.

Problem	No. of	No. of	Time (seconds)				Plan Length				Makespan			
Number	Robots	Goals	orig	max	squ	total	orig	max	squ	total	orig	max	squ	total
1	2	4	0.01	0.01	0.01	0.01	24	24	24	24	24	12	12	12
2	2	4	0	0.01	0.01	0.01	54	54	54	54	54	27	27	27
3	2	4	0.01	0.03	0.02	0.02	74	74	74	74	74	37	37	37
4	2	6	0.02	0.46	13.11	13.07	132	132	153	154	132	66	95	95
5	4	6	0.02	3.91	0.19	0.19	111	174	111	111	111	58	37	37
6	4	8	0.03	–	0.35	0.35	148	–	147	147	148	–	37	37
7	6	8	0.05	–	0.96	0.95	148	–	148	148	148	–	37	37
8	6	10	0.06	–	1.59	1.6	185	–	185	185	185	–	37	37
9	8	10	0.08	–	3.42	3.41	185	–	185	185	185	–	37	37
10	10	10	0.11	–	6.33	6.31	185	–	185	185	185	–	74	74

TABLE III

TABLE SHOWING THE PERFORMANCE OF DIFFERENT VERSIONS OF ADP ON PROBLEM 10 OF SEVERAL IPC DOMAINS. KEY: A IS ADP, A-M IS ADBP-MAX, A-S IS ADBP-SQUARE, AND A-T IS ADBP-TOTAL.

Prob	Search Time (s)						Plan Length						Agent Makespan					
No.	ff	lama	a	a-m	a-s	a-t	ff	lama	a	a-m	a-s	a-t	ff	lama	a	a-m	a-s	a-t
Rovers	0.01	0.02	0.01	0.86	0.19	0.19	37	40	37	37	41	41	13	17	16	17	18	18
Satellite	0.01	0.04	0.02	–	3.58	3.58	35	39	51	–	43	43	14	18	47	–	19	19
Elevators	0.01	0.08	0.01	–	36.28	14.69	29	32	27	–	31	30	15	17	15	–	14	14

a somewhat useful estimate, even when the domain is not symmetrical and not all goals are achievable by all agents. The algorithm is currently at a state where it satisfies its intended role in our application domain, but will need to be improved and updated as the requirements of the environment evolve.

While the algorithm is still at an early stage of development, this paper shows that multiagent heuristic information can be calculated and used effectively in domains previously only solvable by multiagent techniques that can effectively ignore the interactions between agents. This shows that there is perhaps more room than previously thought for investigating different algorithms based on agent decompositions. For example, ADBP currently does not exploit the fact that some actions are not influencing and therefore cannot change the capabilities of the other agents. After a non-influencing action is used to generate a successor state, a reduced heuristic calculation could be performed. We also intend to explore to what extent optimal planning can be employed, and it seems likely it will be possible to find interesting pruning techniques based on the structure afforded by the domain decomposition.

REFERENCES

[1] N. J. Nilsson, "Shakey the robot," AI Center, SRI International, Tech. Rep., 1984.

[2] T. Lozano-Pérez, J. Jones, E. Mazer, and P. A. O'Donnell, "Task-level planning of pick-and-place robot motions," *IEEE Computer*, vol. 22, pp. 21–29, 1989.

[3] S. Bøgh, O. S. Nielsen, M. R. Pedersen, V. Krüger, and O. Madsen, "Does your Robot have Skills?" in *Proceedings of the International Symposium on Robotics (ISR)*, 2012.

[4] R. E. Fikes and N. J. Nilsson, "STRIPS: A new approach to the application of theorem proving to problem solving," *Artificial Intelligence*, vol. 2, no. 3-4, pp. 189–208, 1971.

[5] D. McDermott, M. Ghallab, A. Howe, C. Knoblock, A. Ram, M. Veloso, D. Weld, and D. Wilkins, "PDDL – The Planning Domain Definition Language (Ver. 1.2)," Yale Center for Computational Vision and Control, Technical Report CVC TR-98-003/DCS TR-1165, 1998.

[6] S. Richter and M. Westphal, "The LAMA planner: Guiding cost-based anytime planning with landmarks," *Journal of Artificial Intelligence Research*, vol. 39, pp. 127–177, 2010.

[7] J. Hoffmann and B. Nebel, "The FF Planning System: Fast Plan Generation Through Heuristic Search," *Journal of Artificial Intelligence Research*, vol. 14, pp. 253–302, 2001.

[8] M. Fox and D. Long, "PDDL2.1: An Extension to PDDL for Expressing Temporal Planning Domains," *Journal of Artificial Intelligence Research*, vol. 20, pp. 61–124, 2003.

[9] A. J. Coles, A. I. Coles, M. Fox, and D. Long, "Forward-Chaining Partial-Order Planning," in *Proceedings of the International Conference on Automated Planning and Scheduling*, 2010, pp. 42–49.

[10] A. J. Coles, A. Coles, A. G. Olaya, S. J. Celorrio, C. L. López, S. Sanner, and S. Yoon, "A Survey of the Seventh International Planning Competition," *AI Magazine*, vol. 33, no. 1, pp. 83–88, 2012.

[11] R. I. Brafman and C. Domshlak, "From One to Many: Planning for Loosely Coupled Multi-Agent Systems," in *Proceedings of the International Conference on Automated Planning and Scheduling (ICAPS)*, 2008, pp. 28–35.

[12] R. Nissim and R. I. Brafman, "Multi-agent A* for Parallel and Distributed Systems," in *Proceedings of the International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, 2012, pp. 1265–1266.

[13] D. Borrajo, "Plan sharing for multi-agent planning," in *Proceedings of the ICAPS Workshop on Distributed and Multi-Agent Planning*, 2013, pp. 57–65.

[14] M. Crosby, M. Rovatsos, and R. P. A. Petrick, "Automated Agent Decomposition for Classical Planning," in *Proceedings of the International Conference on Automated Planning and Scheduling (ICAPS)*, 2013, pp. 46–54.

[15] International Planning Competition, "http://www.plg.inf.uc3m.es/ipc2011-deterministic/," 2011.

[16] M. Helmert, "The Fast Downward Planning System," *Journal of Artificial Intelligence Research*, vol. 26, pp. 191–246, 2006.

[17] M. Crosby, "Multiagent Classical Planning," Ph.D. thesis, University of Edinburgh, 2014.

[18] M. Crosby, A. Jonsson, and M. Rovatsos, "A single-agent approach to multiagent planning," in *Proceedings of the European Conference on Artificial Intelligence (ECAI)*, 2014.